# MeLiF+: Optimization of Filter Ensemble Algorithm with Parallel Computing

Ilya Isaev, Ivan Smetannikov

# MeLiF+: Optimization of Filter Ensemble Algorithm with Parallel Computing

Ilya Isaev and Ivan Smetannikov

ITMO University, Computer Science Department,
49 Kronverksky Pr., 197101, St. Petersburg, Russia
isaev@rain.ifmo.ru, ismetannikov@corp.ifmo.ru

**Abstract.** Search of algorithms ensemble – that is, best algorithms combination is common used approach in machine learning. MeLiF algorithm uses this technique for filter feature selection. In our research we proposed parallel version of this algorithm and showed that it is not only improves algorithm performance significantly, but also improves feature selection quality.

**Keywords:** feature selection, variable selection, attribute selection, ensemble learning, feature filters, metrics aggregation, MeLiF, parallel computing

## 1 Introduction

In modern world, machine learning became one of the most promising and studied science areas, mainly, because of its universal application to any data-related problem. One example of such an area is bioinformatics [3; 4; 6; 10], which produces giant amount of data about gene expression of different organisms. This data could potentially allow to determine which DNA pieces are responsible for some visual change of indiviual, or for reactions to particular environment change. The main problem of such data is its huge number of features and relatively low amount of objects. Because of high-dimensional space, it is very hard to build a model which generalizes such data well. Furthermore, a lot of features in such datasets have nothing in common with results, so, they should be treated as noize.

$$A^* = 4$$

It seems to be logical in this case to select somehow the most relevant features and to learn a classifier on these only. This idea is implemented in such area of machine learning as feature selection. There are three main methods of feature selection: filter selection based on statistical measures of every single feature or features subsets, wrapper selection based on subspace search with classifier result as an optimization measure, and embedded selection that uses classificators inner properties [12].

The main peculiarity of filter methods is their speed. This leads to the fact that they are frequently used for preprocessing, and resulting subsets of features further passed to other wrapper or embedded method. This is especially important for bioinformatics, where number of features in datasets is sometimes dozens and hundrends of thousands.

These days, many machine learning algorithms use ensembling [1; 4; 8]. MeLiF algorithm [13] tries to apply this method to feature selection. It builds a linear combination of basic filters, that selects the most relevant features. MeLiF has a structural characteristic that it can be easily modified to work in concurrent or distributed manner. At this research, we implemented parallel version of MeLiF called MeLiF+ and achieved significant speed improvement without losing in selection quality.

The remainder of the paper is organized as follows: MeLiF algorithm is described in Section 2, parallelization scheme is proposed in Section 3, experiment setup and used quality measures are outlined in Section 4, and finally experiment results are contained in Section 5.

## 2 MeLiF

Algorithm treats some linear combinations of basic filters as starting points. It has been observed during experiments that the best option is this following choice of starting points: $(1, 0, ..., 0), (0, 1, ..., 0), ..., (0, 0, ..., 1)$ – only one basic filter matters at the beginning, and $(1, 1, ..., 1)$ – all basic filters are equal at the beginning. Algorithm iterates over the starting points and tries to shift each coordinate value to small constants $+\delta$ and $-\delta$ – value of grid spacing for each point. If some of applied changes succeed, i.e. quality measure for a point after a shift is greater than the maximum value: the algorithm chooses that point and starts searching from its first coordinate. If, all coordinates were shifted to $+\delta$ and $-\delta$ and no quality improvement observed, algorithm stops.

**Algorithm 1** MeLiF algorithm

**Input:** points, delta, evaluate

$q^* \leftarrow 0$
$p^* \leftarrow 0$
**for each** $p \in points$ **do**
    $q \leftarrow evaluate(p)$
    **if** $q > q^*$ **then**
        $p^* \leftarrow p$
        $q^* \leftarrow q$
$smthChanged =$ **true**
**while** smthChanged **do**
    **for each** $dim \in p.size$ **do**
        $p^+ \leftarrow p$
        $p^+[dim] \leftarrow p^+[dim] + delta$
        $q^+ \leftarrow evaluate(p^+)$
        **if** $q^+ > q^*$ **then**

15: $q^* \leftarrow q^+$
16: $p^* \leftarrow p^+$
17: $smthChanged =$ **true**
18: **break**
19: $p^- \leftarrow p$
20: $p^-[dim] \leftarrow p^-[dim] - delta$
21: $q^- \leftarrow evaluate(p^-)$
22: **if** $q^- > q^*$ **then**
23: $q^* \leftarrow q^-$
24: $p^* \leftarrow p^-$
25: $smthChanged =$ **true**
26: **break**
27: **return**$(p^*, q^*)$

Then, for each point obtained during coordinate descent, the algorithm measures value of resulting linear combination of basic filters for each feature in dataset. After that, results are sorted, and the algorithm selects $N$ best features. Then, the algorithm runs some classifier only with that feature subset. The obtained result is saved for comparing with other points and caching. It helps to reduce working time due to visited points usage.

## 3 MeLiF+

We proposed the following improvements to the MeLiF method: each starting point is processed in a distinct thread with global maximum maintained through synchronization point. Moreover, *evaluate* submethod is run concurrently for $+\delta$ and $-\delta$, and selects the best point after retrieving both results. We showed that it not only improves the algorithm performance on multicore system, but also usually improves feature selection quality.

This fact has the following explanation: the original MeLiF algorithm is greedy, so it assumes that if each point it steps in is a local optimum then resulting point will be the global optimum, adding an ability to lookup for two deltas simultaneously allows algorithm to select better local optimum. Also, as starting points are processed in parallel, one thread can find a local optimum. This causes other threads to stop their work even if further descent leads to the better result. This can cause different selection result, better or worse (both cases are presented in Section 5), but experiments show that avarage MeLiF+ results are better.

## 4 Experiments

We used SVM [5] from WEKA [14] library, with polynomial kernel and soft margin parameter $C = 1$ as classifier. To improve stability, we used 5-fold cross-validation. The number of selected features was constant: $N = 100$. In order to compare our method with the old one, we used $F_1$ score [11] of SVM classifier.

As we wanted to know how much our method differs from the original one in terms of space search strategy, we calculated $z$-score for each dataset.

We ran our experiments on a machine with following characteristics: 32-core CPU AMD Opteron 6272 @ 2.1 GHz, 128 GB RAM. We used $N = 50$ threads, $N = 2 \cdot p \cdot f$ threads, where $p$ is the number of starting points, $f$ is the number of folders used for cross-validation.

As basic filters, we used Spearman Rank Correlation (SPC), Symmetric Uncertainty (SU), Fit Criterion (FC) and Value Difference Metric (VDM) [2; 9]. For each dataset, we executed MeLiF and MeLiF+ and stored their working time and points with the best classification result.

We used 50 datasets of different sizes: 33 datasets have been taken from Gene Expression Omnibus, 5 from Kent Ridge Bio-Medical Dataset, 5 from RSCTC'2010 Discovery Challenge, 4 from Broad institute Cancer Program Data Sets, 3 from Feature Selection Datasets at Arizona State University. Some datasets were multi-labeled, therefore we splitted them into several derivative binary datasets with commonly used one-versus-all technique. Then we excluded datasets that contained too few instances of one of the classes. After that, we used standard feature scaling and discretized all features to 11 different values from -5 to 5.

## 5 Results

Table below contains experiment results. All the datasets are sorted by their total size which is basically a multiplication of their features and objects number. In $F_1$ score comparison of MeLiF and MeLiF+ better results for each dataset are highlighted in grey, equal results are not highlighted. Runtime is presented in seconds. At the last column, z-score is provided.

Table 1: MeLiF in comparison with swarm algorithms

| Dataset | Size | $F_1$ score | | Time | | z-score |
|---|---|---|---|---|---|---|
| | | MeLiF | MeLiF+ | MeLiF | MeLiF+ | |
| SRBCT30 | 191k | 0.900 | 0.891 | 13 | 2 | 0.23 |
| SRBCT31 | 191k | 1.000 | 1.000 | 17 | 3 | 0 |
| GDS2960 | 417k | 0.971 | 0.980 | 33 | 7 | -10.98 |
| CNS | 427k | 0.742 | 0.791 | 33 | 6 | -1.06 |
| Leuk3c0 | 513k | 0.986 | 0.986 | 34 | 5 | 0 |
| Leuk3c1 | 513k | 0.933 | 0.933 | 33 | 5 | 0 |
| GDS2961 | 566k | 0.845 | 0.845 | 49 | 13 | 0 |
| GDS2962 | 566k | 0.784 | 0.887 | 45 | 11 | -11.15 |
| DLBCK | 962k | 0.799 | 0.734 | 65 | 13 | 4,67 |
| GDS2901 | 1337k | 1.000 | 1.000 | 88 | 17 | 0 |
| Prostate | 1713k | 0.925 | 0.903 | 93 | 34 | 7.27 |
| GDS4109 | 1760k | 0.936 | 0.936 | 142 | 38 | 0 |
| GDS5083 | 2131k | 0.862 | 0.862 | 195 | 60 | 0 |

| | | | | | | |
|---|---|---|---|---|---|---|
| d2t0 | 2339k | 0.765 | 0.765 | 172 | 71 | 0 |
| d2t1 | 2339k | 0.779 | 0.844 | 180 | 60 | -9.73 |
| breast | 2346k | 0.769 | 0.812 | 161 | 24 | -11.71 |
| GDS4261 | 2367k | 1.000 | 1.000 | 130 | 16 | 0 |
| GDS3257 | 2384k | 0.980 | 1.000 | 131 | 17 | -33.65 |
| GDS4901 | 2515k | 0.899 | 0.946 | 220 | 60 | -7.6 |
| GDS3553 | 2543k | 1.000 | 1.000 | 142 | 18 | 0 |
| GDS3116 | 2584k | 0.826 | 0.826 | 142 | 23 | 0 |
| GDS4336 | 2598k | 0.865 | 0.865 | 200 | 66 | 0 |
| GDS5047 | 2925k | 0.989 | 0.989 | 185 | 41 | 0 |
| GDS3995 | 3200k | 1.000 | 1.000 | 181 | 24 | 0 |
| GDS496840 | 3296k | 0.890 | 0.890 | 226 | 40 | 0 |
| GDS496841 | 3296k | 0.953 | 0.936 | 224 | 40 | 4.04 |
| GDS2947 | 3499k | 1.000 | 1.000 | 217 | 28 | 0 |
| GDS43181 | 3591k | 0.913 | 0.913 | 275 | 64 | 0 |
| arizona5 | 3738k | 0.730 | 0.754 | 219 | 67 | -7.47 |
| Ovarian | 3833k | 1.000 | 1.000 | 192 | 23 | 0 |
| GDS4103 | 4264k | 0.918 | 0.918 | 265 | 71 | 0 |
| GDS2771 | 4265k | 0.760 | 0.760 | 299 | 81 | 0 |
| GDS503730 | 4428k | 0.738 | 0.826 | 243 | 140 | -16.77 |
| GDS503732 | 4428k | 0.782 | 0.782 | 293 | 69 | 0 |
| GDS3929 | 4488k | 0.821 | 0.821 | 376 | 74 | 0 |
| GDS483731 | 4811k | 0.920 | 0.920 | 413 | 130 | 0 |
| GDS483733 | 4811k | 0.951 | 0.951 | 316 | 48 | 0 |
| d5t | 4860k | 0.869 | 0.869 | 370 | 70 | 0 |
| GDS3622 | 4961k | 1.000 | 1.000 | 266 | 33 | 0 |
| GDS2819 | 5412k | 0.991 | 1.000 | 436 | 149 | -4.05 |
| d6t | 5428k | 0.792 | 0.792 | 381 | 69 | 0 |
| GDS413O | 5685k | 1.000 | 1.000 | 315 | 39 | 0 |
| d4t | 6178k | 0.719 | 0.719 | 513 | 124 | 0 |
| GDS4129 | 6561k | 1.000 | 1.000 | 354 | 43 | 0 |
| GDS4222 | 7107k | 0.965 | 0.971 | 454 | 84 | -7.42 |
| GDS4431 | 7973k | 0.802 | 0.802 | 537 | 100 | 0 |
| arizona1 | 8847k | 0.823 | 0.823 | 558 | 85 | 0 |
| GDS4600 | 9294k | 0.979 | 0.968 | 472 | 124 | 23.27 |
| GDS3244 | 9787k | 1.000 | 1.000 | 505 | 65 | 0 |

As it can be seen from the table above, MeLiF+ is always at least 3 times faster than the MeLiF, and this difference gets up to 6 times for some datasets. Although MeLiF and MeLiF+ have almost the same results in $F_1$ score, there is some difference in their work on 15 datasets as provided via $z$-score. But only in 5 cases MeLiF+ had worse results than original the MeLiF algorithm. But on 36 datasets, they performed equally and at 11 datasets new algorithm outperformed the original one.

## 6 Conclusion

The proposed parallelization scheme made algorithm in average to work 5.5 times faster without affecting selection quality. Unforunately, in this research we did not achieved linear speed improvement because of the fixed maximum of parallel processed points. In our future work, we are planning to use threads pool which is limited by the testing system and achieve linear speed growth with using exploration and exploitation [7] strategy to spread the search points in the search space. Also this should lead to high increase in optimized measure.

## 7 Acknowledgements

Authors would like to thank Julia Ugarkina and Andrey Filchenkov for useful comments and proofreading. This work was financially supported by the Government of Russian Federation, Grant 074-U01.

# Bibliography


[1] Abeel, T., Helleputte, T., Van de Peer, Y., Dupont, P., Saeys, Y.: Robust biomarker identification for cancer diagnosis with ensemble feature selection methods. Bioinformatics 26(3), 392–398 (2010)

[2] Auffarth, B., López, M., Cerquides, J.: Comparison of redundancy and relevance measures for feature selection in tissue classification of ct images. In: ICDM. pp. 248–262. Springer (2010)

[3] Bolón-Canedo, V., Sánchez-Maroño, N., Alonso-Betanzos, A., Benítez, J., Herrera, F.: A review of microarray datasets and applied feature selection methods. Information Sciences 282, 111–135 (2014)

[4] Bolón-Canedo, V., Sánchez-Maroño, N., Alonso-Betanzos, A.: An ensemble of filters and classifiers for microarray data classification. Pattern Recognition 45(1), 531–539 (2012)

[5] Burges, C.J.: A tutorial on support vector machines for pattern recognition. Data mining and knowledge discovery 2(2), 121–167 (1998)

[6] Chuang, L.Y., Yang, C.H., Wu, K.C., Yang, C.H.: A hybrid feature selection method for dna microarray data. Computers in biology and medicine 41(4), 228–237 (2011)

[7] Desautels, T., Krause, A., Burdick, J.W.: Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization. The Journal of Machine Learning Research 15(1), 3873–3923 (2014)

[8] Dietterich, T.G.: Ensemble methods in machine learning. In: Multiple classifier systems, pp. 1–15. Springer (2000)

[9] Filchenkov, A., Dolganov, V., Smetannikov, I.: Pca-based algorithm for constructing ensembles of feature ranking filters. In: proceedings of ESANN conference. pp. 201–206 (2015)

[10] Haury, A.C., Gestraud, P., Vert, J.P.: The influence of feature selection methods on accuracy, stability and interpretability of molecular signatures. PloS one 6(12), e28210 (2011)

[11] Huang, H., Xu, H., Wang, X., Silamu, W.: Maximum f1-score discriminative training criterion for automatic mispronunciation detection. Transactions on Audio, Speech, and Language Processing 23(4), 787–797 (2015)

[12] Saeys, Y., Inza, I., Larrañaga, P.: A review of feature selection techniques in bioinformatics. bioinformatics 23(19), 2507–2517 (2007)

[13] Smetannikov, I., Filchenkov, A.: MeLiF: Filter Ensemble Learning Algorithm for Gene Selection. In: Advanced Science Letters. p. to appear. American Scientific Publisher (2016)

[14] Waikato, T.U.: Weka 3: Data Mining Software in Java. `http://www.cs.waikato.ac.nz/ml/weka/` (2016), [Online; accessed 7-May-2016]